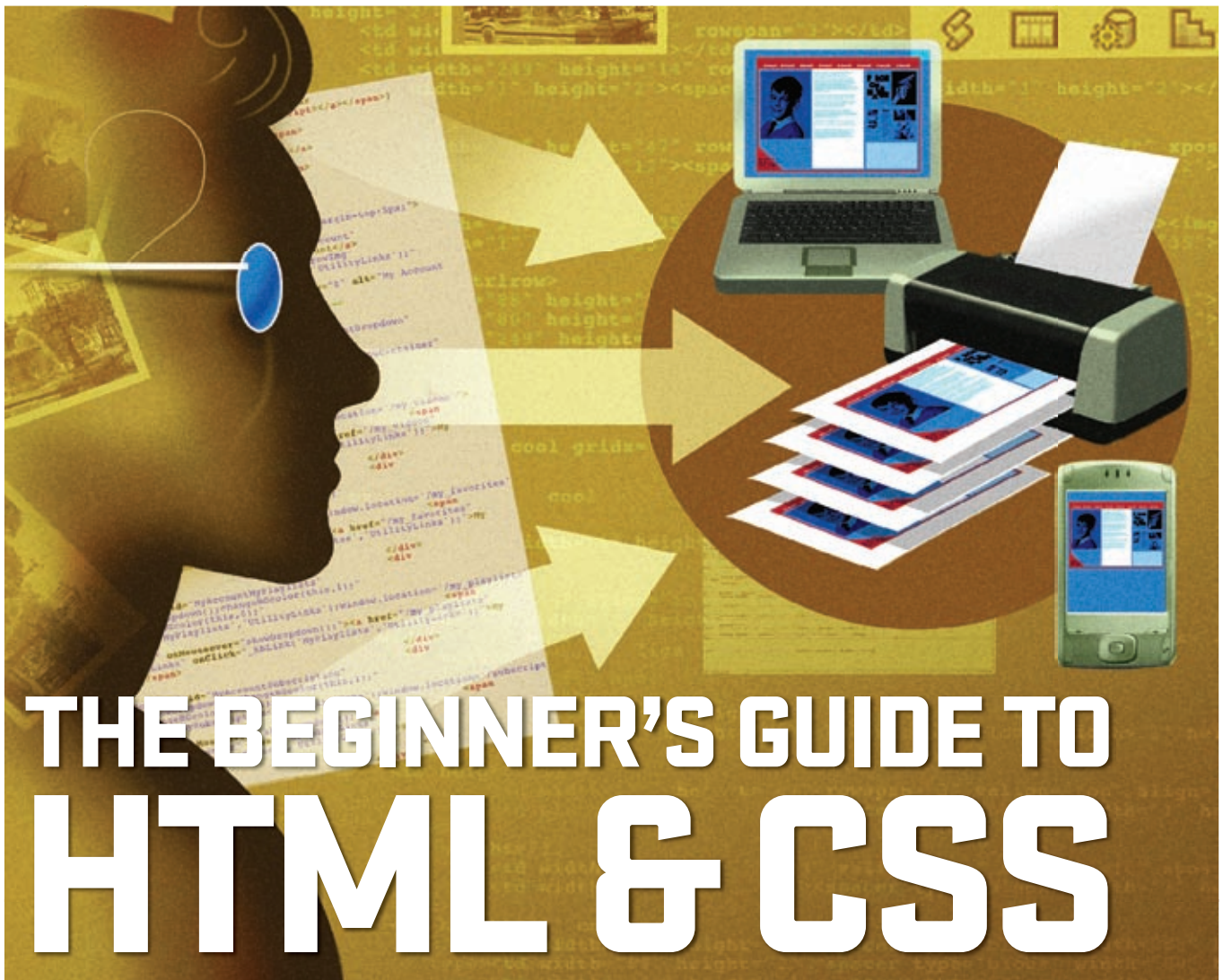




THE BEGINNER'S GUIDE TO HTML & CSS

In the second part of our series on creating your own website, Nik Rawlinson explains how to tailor your pages so they can be viewed on a PDA and as a perfect printout

In last month's *Computer Shopper* we created a small website consisting of three files: an image, a homepage and a stylesheet. The homepage contained the text of our site, while the clever stuff – the formatting and layout – was placed in the stylesheet. This makes it easier to change the formatting when we need to use the page in different ways, which we'll consider here. If you missed last month's tutorial, you'll find it on our cover disc. The stylesheet and homepage referenced here are contained in that tutorial.

The stylesheet currently attached to our homepage is so simple it's suitable only for use on a conventional computer monitor. It's too tall and wide and the fonts are too large to be successfully displayed on a PDA. It also includes an illustrative photo, which is nice but redundant. If you wanted to find directions to the party the site was advertising, you'd have to click away from the front page; the general blurb and the specifics are kept separate. This is fine online, where you can click backwards and forwards at will. However, when you're printing the invitation to take with you, you need a single page, and if you want to download the invite to your PDA, you need something that fits clearly on the small screen.

Here we'll create two new stylesheets for use on a printout and a PDA. We'll also make one small change to our homepage to remove the direct reference to the embedded image file. This way, the content of the virtual on-page box will be defined by the medium used to view the page. We'll then show you some clever tricks that will enable you to generate pages that have different content when printed to what they display onscreen.

UNDERSTANDING THE THEORY

Let's start with some practical theory, which will demonstrate why the content and coding are in separate linked files. Duplicate the styles.css file and keep the original in a safe place. Open your copy and make a few simple changes to the font tags you'll find inside. We want to make substantial changes so you can see the difference they make. Edit the .header section as follows:

```
.header {  
  font-family: Georgia, "Times New  
    Roman", Times, serif;  
  font-size: 20px;  
  font-weight: normal;  
  color: #000000;  
}
```

```
height: 20px;  
top: 1px;  
}
```

Save this and reload the index.html file, making sure both files are in the same folder. You'll see we have changed the font face and size, weight and colour. It was originally a sans serif font, heavy, red, 24 point; it's now black, serif, 20 point, regular weight.

This is why good modern web design practice is to keep design and content separate from one another. A programmer can work on the layout while a dedicated editorial person handles content. Even if you're a solo developer working on your personal homepage, there are benefits to working this way. Your site will be more compliant to current design conventions. If you stick to redefining standard HTML tags such as <h1> (header), <a> (anchor) and <p> (paragraph), visually impaired visitors can also apply their own custom stylesheets to your website to make pages easier to read.

Furthermore, you may find your pages load quicker when the net is particularly congested. Many browsers will display a page without the stylesheet if it experiences any delay loading and then apply the formatting information a few



▲ By including a printer-specific stylesheet, you can produce two versions of a single page. The visitor is presented with the one most appropriate to the medium through which they are reading your content

seconds later if and when it becomes available. By working this way, there's a greater chance your visitors will be able to read what they are interested in – the raw content of your page – even if it doesn't quite look the way you want.

We are not restricted to just changing the style of our text and images through the stylesheet. We can also change the physical layout. Skip to the .container section in the styles.css file and change the positional information to:

```
.container {  
    width: 420px;  
    top: 20px;  
    margin-right: auto;  
    margin-left: auto;  
}
```

This moves the whole content of our page further down the screen.

By making two small changes – and without touching the index.html page – we have changed what our visitors see without risking introducing grammatical, spelling or other errors into the existing text.

Now that you've seen how the stylesheet and homepage relate to each other and how a change to one affects the other, let's reinstate the original styles.css and start work in earnest.

FORMATTING FOR PRINT

Some of the content of our homepage, such as links, is irrelevant when it's not being viewed live and online; it's navigation, not information. We want to make sure it doesn't appear on the printed version. More importantly, we don't want our visitors to have to select the new layout manually when they come to print our page, as print-friendly version links are old hat, sloppy and lazy. We'll add a line to our homepage to handle this, but for now create a new file called print.css containing the following code:

```
.details {  
    height: 0px;  
    visibility: hidden;  
}  
.directions {  
    text-align: left;
```

```
font-family: Georgia, "Times New  
    Roman", Times, serif;  
font-size: 12pt;  
}  
.header {  
    font-family: Arial, Helvetica,  
    sans-serif;  
    font-size: 24px;  
    font-weight: bold;  
}  
.navigation {  
    visibility: hidden;  
}
```

This stylesheet is much shorter than the one we defined for screen display, because positional statements are no longer so important. What we want is a simple top-down page that is easy to follow when away from our PC.

The page links displayed on the left-hand side of our original page are useless when it's viewed by any means other than a browser, so we have suppressed the frame that contains them by stripping its formatting information down to:

```
visibility: hidden;
```

This hides the whole layer. Other lines stripped out include the body style tag for centring our content. This was a workaround that forced Internet Explorer 6 to centralise everything on the page, as that browser ignores our automated margin setting. Fortunately, it isn't required for Firefox and other Mozilla-based browsers, which interpret our margin-right: auto; and margin-left: auto; codes correctly.

The remaining lines should be easy to understand, as they are derived from the original styles.css file we created in the last issue.

FORMATTING FOR PDA

We'll perform a similar hatchet job on the PDA stylesheet, removing most of our positional commands so the page runs in a simplified, linear fashion from top to bottom.

Here we'll see the second benefit of defining much of our non-text page content in an attached stylesheet rather than the page itself:

bandwidth conservation. We are attaching three stylesheets to our page rather than the one used in the original incarnation. This means we'll be downloading some information that will never be used, but we are only going to import our images if they are actually used. As such, the bandwidth-gobbling photo is downloaded only by someone using a full-blown browser on a proper computer. PDA users save time and, if accessing over GPRS, money by downloading a smaller map fragment.

Call this stylesheet pda.css and enter:

```
.directions {  
    visibility: hidden;  
}  
.header {  
    font-family: Arial, Helvetica,  
    sans-serif;  
    font-size: 12px;  
    font-weight: bold;  
}  
.navigation {  
    visibility: hidden;  
}  
.details {  
    text-align: left;  
    font-family: Georgia, "Times New  
    Roman", Times, serif;  
    font-size: 10pt;
```

There is one difference between this stylesheet and the one we'll use for printed output: the size of the font in the header. This has been reduced to 12px (pixels) so it won't spread across two lines and is for aesthetic purposes only.

If we wanted to use either of these stylesheets in preference to the one already linked to our page, we would save it with the same name as our existing sheet (styles.css), and the page would be reformatted without us touching the underlying code. However, as we have specific uses for both these stylesheets we'll go back to the index.html file and embed them, with media attributes determining where and when each one is used.

EMBEDDING OUR CHANGES

Open index.html, the file we created in the first part of this walkthrough, and look at the top section between the <head> and </head> tags. This is where we have embedded the styles.css stylesheet that controls the onscreen look of our page. Change everything between the two <head> tags so it says:

```
<head>  
<title>Computer Shopper Party</title>  
<link href="styles.css"  
    media="screen" rel="stylesheet"  
    type="text/css">  
<link href="print.css" media="print"  
    rel="stylesheet" type="text/css">  
<link href="pda.css" media="handheld"  
    rel="stylesheet" type="text/css">  
</head>
```

As you can see from the code, different stylesheets will be applied depending on the medium (computer screen, PDA or printed page) used to access the content. Any user clicking their browser's print button will end up with a simplified version of the page without the links or associated photo. Best of all, they won't



have to think first about specifying that they want a printer-friendly page, because we've already done the hard work for them.

We do need to make one further change to this page. We need to specify a second image file for use in printed output and anything displayed on a PDA. Scroll down the code to find:

```
<div id="photo" class="photo"></div>
```

Immediately below it, add:

```
<div id="map" class="map"></div>
```

We now have two embedded images: the photo we have been using, and a map showing where our guests need to go. However, we don't want them both to display. Instead, we want to show just the photo onscreen, and the map for the PDA and printed output.

Open `styles.css` and add a new class for `.map` as follows:

```
.map {
  visibility: hidden;
  height: 0px;
}
```

We need to perform a similar edit inside our other stylesheets, but this time to hide the original photo. Open both `pda.css` and `print.css` and add:

```
.photo {
  visibility: hidden;
  height: 0px;
}
```

```
.map {
  border: 10px solid #000000;
  margin-top: 10px;
  margin-bottom: 10px;
  height: 300px;
  width: 300px;
}
```

We've made significant improvements to the printed output of our page, but it's still not perfect. It doesn't really need the introductory blurb shown on the screen version. Let's re-use the layer-hiding trick we applied to the navigation panel to produce different pages for print and screen. We might display only the barest details on the page itself, but the printed version should present our visitor with a complete invitation, with the map supplemented by written directions. The PDA edition, meanwhile, will continue to display only a subset of the information displayed on a regular full-sized computer screen.

NEW DIRECTIONS

To achieve this, we'll amend the `index.html` file to add a new layer called `directions`, which, like the other layers on our page, is styled up using the three attached stylesheets.

Start a new line in `index.html` just below the line that describes our photo layer and add the following:

```
<div id="directions"
class="directions">Take the tube
to Great Portland Street station
(Circle, Metropolitan or Hammersmith
& City Line), and exit to street
level. Cross the road towards Tesco
Metro and turn left. Pass the Green
Man pub and take first road on right
(Cleveland Street). Follow this for
```

```
around five minutes, crossing over
a cross-roads at traffic lights. We
are around 100m further on; a white
building on the left.</div>
```

The `&` code in the middle of this paragraph invokes an ampersand. The reason we can't just use the ampersand symbol on its own is that it is used as a pointer for descriptive code used to expand a standard character set. This is necessary because we can't be sure which characters are available in the standard font being used by our visitors. Another important and useful code for British users is `£`; which is used for a £ sign.

We need to define which versions of our page this paragraph should appear in. Open the three stylesheets. At the top of each file add:

```
.directions {
  visibility: hidden;
  height: 0px;
}
```

Replace the details section in the `print.css` file with:

```
.details {
  visibility: hidden;
  height: 0px;
}
.directions {
  text-align: left;
  font-family: Georgia, "Times New
  Roman", Times, serif;
  font-size: 12pt;
}
```

Upload everything to your server and test using a regular browser on your PC or Mac, printed output and a PDA if you have network access.

Feed frenzy Adding dynamic data to your website

Your website may need to include constantly updated information to keep visitors abreast of changes. In our example, travel news is an important consideration as it has a bearing on the time your visitors leave home.

Fortunately, there is a wide range of information sources freely available online. The BBC, for example, publishes a range of data feeds in XML format that can be quickly and easily incorporated into your site, and even included in your printed output. However, the BBC's terms and conditions state that this information can be used only for personal purposes and must never be exploited commercially.

When you visit a site with an associated feed, you'll see a small feed icon. Usually this will be an orange square with rounded corners and three white arcs radiating out from the bottom left-hand corner. This links to the feed address that you need to copy. If you can't work out how to extract the feed address, view the page content and search for a URL with the extension `.xml`, `.rss` or `.rdf`. The feed for the BBC's London Underground transport travel status news, for example, is at www.bbc.co.uk/travelnews/tpg/en/pti/tube_rss.xml. Like any XML feed, the data this spits out is unusable in its raw format, so

it has to be parsed – processed into a readable format – before being displayed on your page. You need to publish your pages in PHP format rather than simple HTML, but this is no problem if your web host provides PHP. Simply change the extension of your existing `index.html` file to `.php`.

If you're happy to have a small advert displayed at the bottom of your feed, we recommend using the free parsing tool at www.site-reference.com/rss-parser.php. You need only enter your site address and the URL of the feed to have the necessary code snippet generated, ready for copying and pasting to your site. Whether you choose to put this on

your screen-displayed homepage or the printout is up to you. We recommend attaching it to the bottom of the `directions` paragraph inside the `.directions` layer. To see how this works, check the feed running down the fourth column of www.nik.co.uk.

Once you have incorporated your feed, preview your page to see how the information is styled. If it jars with your existing site design, use your stylesheets (`styles.css` if you are displaying the information onscreen, `print.css` if you have embedded it in the printed edition) to change its appearance so it matches what you already have in place.

```
<?xml version="2.0"?>
<channel>
  <title>
    BBC Travel News | Top incidents for The Tube (All Lines)
  </title>
  <link>
    http://www.bbc.co.uk/travelnews/public/tube/tube.shtml
  </link>
  <description>
    Updated traffic and travel information, around the clock, around the country
  </description>
  <language>en-gb</language>
  <lastBuildDate>Mon, 6 Nov 2006 17:32:54 GMT</lastBuildDate>
  <copyright>Copyright: (C) British Broadcasting Corporation</copyright>
  <image>
    <title>BBC Travel News</title>
```

XML data feeds aren't very friendly when viewed in their raw state, so they must be parsed before being embedded into a page. The resulting data can be formatted using your existing stylesheets